

Capítulo 1

LaTeX 2_ε

¿Qué necesita para ser funcionar LaTeX 2_ε?
—What does LaTeX 2_ε need to work?

Este capítulo establece los recursos necesarios para el desarrollo de documentos con el sistema LaTeX 2_ε. Esta trata los Fundamentos, Materiales e instalación, Montaje y Procesamiento del primer documento, Procedimiento sintaxis del lenguaje; todo lo necesario para arrancar desde un punto cero, (i.e., una introducción completa) técnicamente hablando.

1.1. Fundamentos

1.1.1. Versión

TeX (Typesetting system) 1977 (inicio), 1982 (lanzamiento), 1989 (última mejora).

Desarrollado por Donald Ervin Knuth en 1982 como "un nuevo sistema de composición tipográfica [—typesetting system] para la creación de libros hermosos —y especialmente para libros que contienen mucha matemática." [1].

Actualmente es una marca registrada de la American Mathematical Society (AMS).

LaTeX 2.09 (Document Preparation System) 1985-09-01.

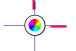
Desarrollado por Leslie Lamport en 1985 [2] (inicialmente llamado *Document Preparation System*) para ser un lenguaje de alto nivel, que internamente reduce el código al lenguaje de bajo nivel TeX.

LaTeX 2_ε (A mathematical typesetting language) 1994-06-01 to 2024-11-01.

Desarrollado por el *LaTeX Project Team* desde 1994 [3] para ser el nuevo estándar LaTeX del Siglo 21.

LaTeX3 (The long-term future of LaTeX) 2009 (inicio), 2017-12-16 to 2021-02-18.

Desarrollado por el *LaTeX3 Project Team* desde el 2009 [4] (llamado *El futuro a largo-plazo de LaTeX*) para ser el nuevo lenguaje de programación de paquetes y clases para LaTeX.



1.1.2. Cualidades

- Software libre
- Independiente de la plataforma, la salida es la misma en todas las plataformas
- Permite el control más fino sobre cualquier aspecto tipográfico del documento
- No es un procesador de texto WYSIWYG (“what you see is what you get”), sino un lenguaje de programación interpretado.
- Diseñado para escribir el documento según el diseño lógico YAFIYGI (“you asked for it; you got it”) a diferencia de WYSIWYG de diseño visual.

1.1.3. Plataforma

La plataforma de desarrollo basicamente esta compuesto por:

- Motor —Composition engine
- Lenguaje —Language programming
- Bibliotecas —Packages

Una distribución de TEX trae consigo el motor y los paquetes básicos. Distribuciones de TEX:

MiKTeX miktex.org/download
(Windows) A separate distribution entirely that

TeX Live www.tug.org/texlive/
(Windows, Linux, and OS X), the standard, cross-platform distribution.

MacTeX www.tug.org/mactex/
(Mac OS X) A packaged version of TeX Live made for OS X with some Mac-specific tools

1.2. Entorno de desarrollo

1.2.1. Instalación del sistema

La Figura 1.1 muestra la documentación de la distribución MiKTeX (miktex.chm→P.1035) [5] y TeX Live (readme.es.html→P.1035) instaladas.

1.2.2. Programas de administración

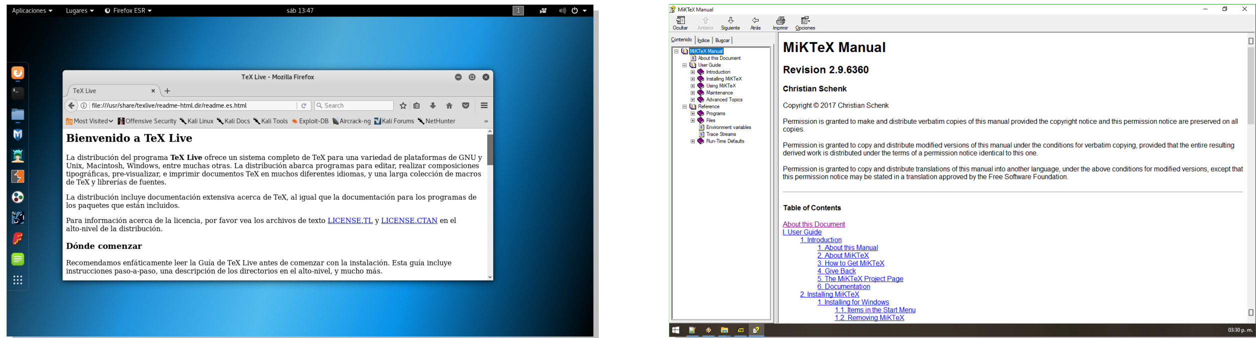
Los programas de administración y configuración del sistema son:

- `tlmgr`→P.4, administrador de TeX Live;
- `mo.exe`→P.5, administrador de MiKTeX.

Engine `tlmgr` ⟨*opciones*⟩ ⟨*acción*⟩ ⟨*option*⟩ ⟨*operand*⟩
TeX Live Manager. Este programa de consola es el administrador de opciones de TeX Live.
Alguna de las ⟨*acciones*⟩ son:

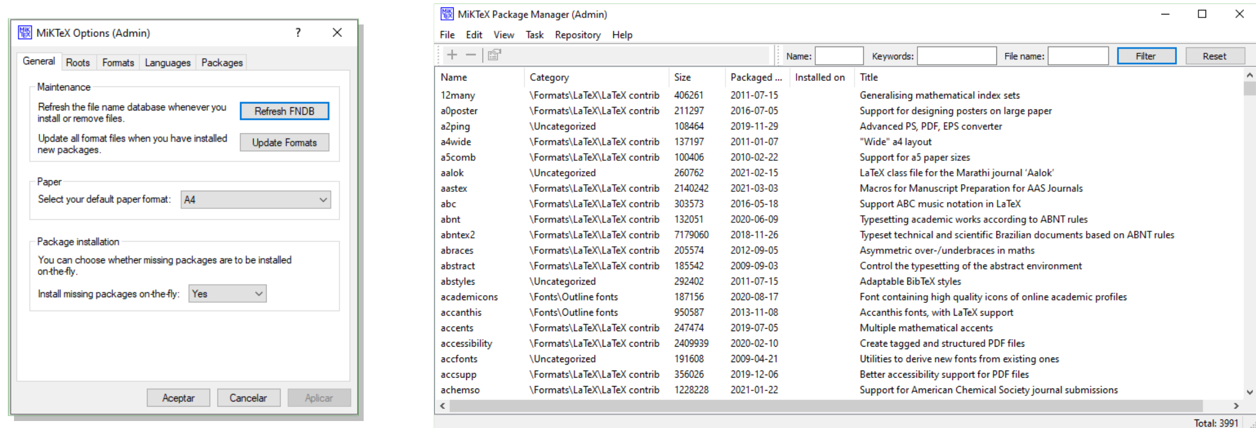
- `--version`

Figura 1.1. Documentación de la distribución instalada



a). Linux – TeX Live 2018 instalado **b).** Windows – MiKTeX 2.9 instalado

Figura 1.2. MiKTeX – Maintenance



a). MiKTeX Options (Admin) **b).** MiKTeX Package Manager (Admin)

- `--help`

```
# tlmgr --version
(running on Debian, switching to user mode!)
tlmgr revision 47951 (2018-06-07 07:55:43 +0200)
tlmgr using installation: /usr/share/texlive
TeX Live (http://tug.org/texlive) version 2018
```

Engine `mo.exe`

MiKTeX Options. Este programa ejecutable (mo.exe→P.1035, mo_admin.exe) [5, p. 15] es el administrador de opciones generales de MiKTeX.

En la Figura 1.2 el **a)** es el administrador de opciones de MiKTeX que se puede invocar via: `mo_admin`.

En la Figura 1.2 el **b)** es el administrador de paquetes que se puede invocar vía: `mpm_mfc_admin` Instalación de paquetes automática.

1.2.3. Instalación del IDE

Desarrollo integrado (IDE): Editores especializados para código LaTeX [6, p. 5] que ayudan en el desarrollo:

Capítulo 2

Interfaces fundamentales

Interfaces de composición tipográfica
—A TEX programming environment

Este capítulo establece los conocimientos fundamentales para empezar a escribir código en los lenguajes con base TEX, iniciando en como se interpreta un carácter del código fuente, su composición como símbolo renderizado y posterior ubicación en el texto.

2.1. El Manejo del texto

2.1.1. Terminología

Los términos *carácter*, *glifo* y *punto de código* se usan según lo definido por Unicode [14, §1.3].

2.1.2. Carácter

Un *carácter* es una unidad atómica de texto [15, §11.1.1].

`\char`⟨*number*⟩

Comando primitivo [1, p. 286, 43], expande el glifo del carácter ⟨*number*⟩ de la fuente actual.

⟨*number*⟩ es un número entero de 8-bit's (0 a 255), si excede esta capacidad se produce un error (`! Bad character code`) en tiempo de compilación.

Por ejemplo, el carácter '@' (U+0040, decimal 64):

@	`\char64`

Un *carácter de control* puede ser escrito con la secuencia:

`^^`⟨*tokens*⟩

Esta secuencia representa un carácter de control. [1, p. 368]

Donde ⟨*tokens*⟩ puede ser: *una letra mayúscula* (Ctrl-letter), del código de ASCII; o dos *dígitos hexadecimales*.

Por ejemplo, `^^M`→P.27 y `^^0d` es el carácter de control CR (U+000D).

PRINT PREVIEW (BETA TEST) ~ PRESENTACIÓN PRELIMINAR

13, 13	`\number`\^^M,` `\number`\^^0d`

`\number`⟨*number*⟩

Comando primitivo [1, p. 213, 472], expande la representación base decimal de ⟨*number*⟩→P.228.

64	``\number`@``

918000	`\number"E01F0`

2.1.3. Código de categoria

`\catcode`⟨*carácter*⟩=⟨*categoria*⟩

Comando primitivo TeX, establece la ⟨*categoria*⟩ del ⟨*carácter*⟩.

⟨*carácter*⟩: es un número entero de 8-bits, véase `\char`→P.21.

⟨*categoria*⟩: es un número entero entre 0 a 15, véase Tabla 2.1. Para expandir el *código* [1, p. 271] [16, p. 82] de ⟨*categoria*⟩ puede usar `\number`→P.22 o `\the`→P.250.

12	`\number\catcode64`

La clasificación del ⟨*carácter*⟩ es vigente con visibilidad precedente en el grupo local.

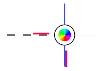
@ es de categoria 11

```
{
    \catcode64=11
    \char64\ es de categoria \the\catcode64
}
```

La Tabla 2.1 (latex.ltx→P.1035, Ln: 282) [8, p. 14] [1, p. 37] muestra la clasificación de los caracteres, con el comando `\catcode` en 16 categorias numeradas del 0 al 15.

Tabla 2.1. Códigos de categoria

Código	Categoria	Carácter	Plain TeX	Unicode
0	Carácter de escape	\		005C
1	Inicio de grupo	{		007B
2	Fin de grupo	}		007D
3	Cambio matemático	$		0024
4	Alineador de tabulación	&		0026
5	Fin de línea	CR	^^M	000D
6	Parámetro de macro	#		0023
7	Exponente	^		005E
8	Subíndice	_		005F
9	Carácter ignorado	NUL	^^@	0000



Capítulo 3

Composición matemática

Composición tipográfica para escribir matemáticas
—Typesetting standard math

3.1. Modo matemático

3.1.1. Math mode

`$`*x*`$` (group operators)

Token '`$`' [1, p. 457] (en la apertura) pasa al modo matematico o (en el cierre) retorna al modo previo. x son los tokens a componer en modo matemático. En este modo los caracteres de espacio estándar (véase `\space`→P.29) se ignoran y los espacios deben ingresarse explícitamente (con un token o comando definido para ello).

Lorem x^2 ipsum	`Lorem $ x^2 $ ipsum`
xy vs. $x\ y$	`$ x y $ vs. $ x\ y $`

`\(`⟨*formula*⟩`\)`

Comandos robustos [8, p. 275], es igual a `$`→P.171 pero, si se usa en modo matemático se produce un error (`! LaTeX Error: Bad math environment delimiter.`).

Lorem x^2 ipsum	`Lorem \(x^2\) ipsum`

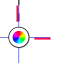
`\math`
⟨*contenido del entorno*⟩
`\endmath`

Macros [8, p. 277] alias de `\(`→P.171 y `\)`.

`\begin{math}`
⟨*contenido del entorno*⟩
`\end{math}`

Versión entorno de `\math`→P.171.

Lorem x^2 ipsum	`Lorem \begin{math}x^2\end{math} ipsum`



PRINT PREVIEW (BETA TEST) ~ PRESENTACIÓN PRELIMINAR

\ensuremath⟨*formula*⟩

En modo párrafo expande $⟨*formula*⟩$ y en math mode expande ⟨*formula*⟩ [8, p. 279] [2, p. 53, 187].

Lorem x^2 ipsum	`Lorem \ensuremath{x^2} ipsum`

3.1.2. Display math mode

$$*x*$$ (group operators)

Token [1, p. 457] para componer la formula x en modo presentación.

Lorem ipsum dolor sit amet, $$e^{i\pi} = 1$$ consectetuer adipiscing elit.	`Lorem ipsum dolor sit amet,`
	`$$ e^{i\pi} = 1 $$`
	`consectetuer adipiscing elit.`

\[⟨*formula*⟩\]

Comandos robustos [8, p. 276, 280]; si esta en modo vertical expande `\nointerlineskip\makebox[.6\linewidth]{}`; luego es igual a `$$` pero, si se usa en modo matemático se produce un error (`! LaTeX Error: Bad math environment delimiter.`).

$$e^{i\pi} = 1$$	`\[ e^{i\pi} = 1 \]`

Extendido. El paquete `amsmath`→P.403 lo redefine como alias de su entorno `equation*`→P.406.

\displaymath
⟨*contenido del entorno*⟩
\enddisplaymath

Macros [8, p. 277] expanden `\[`→P.172 y `\]\@ignoretrue` respectivamente.

\begin{displaymath}
⟨*contenido del entorno*⟩
\end{displaymath}

Versión entorno [2, p. 188] de `\displaymath`→P.172.

$$e^{i\pi} = 1$$	`\begin{displaymath}`
	`e^{i\pi} = 1`
	`\end{displaymath}`

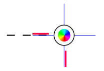
\lefteqn⟨*formula*⟩

Macro [8, p. 279] expande a `\rlap{$\displaystyle #1$}`; i.e., expande la ⟨*formula*⟩, en modo presentación, a la derecha del punto actual.

Lorem. $\sqrt{\frac{1}{2}}$ ipsum (superpuesto)	`Lorem.\lefteqn{\sqrt\frac12}ipsum`

3.1.3. Every math mode

\everymath={⟨*formulas*⟩} (initially {})



Capítulo 4

Composición gráfica

Composición de gráficos vectoriales
—Standard picture environment

4.1. Modo gráfico

\begin{picture}(⟨w,h⟩)(⟨x_o,y_o⟩)
  ⟨*contenido del entorno*⟩
\end{picture}

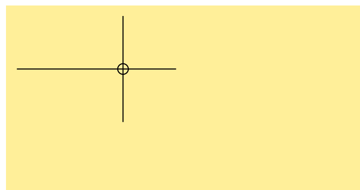
Reserva un rectángulo invisible con base ⟨w⟩ y altura ⟨h⟩.

Origen del sistema: esquina inferior isquierda.

⟨x_o,y_o⟩: Asigna coordenadas arbitrarias a la esquina inferior isquierda.



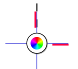
```
\begin{picture}(60,40)
    \put (0,0){\line(0,1){40}}
    \put (0,0){\line(1,0){60}}
    \put (0,0){\circle{4}}
\end{picture}
```



```
\begin{picture}(60,40)(-40,-20)
    \put (0,-20){\line(0,1){40}}
    \put (-40,0){\line(1,0){60}}
    \put (0,0){\circle{4}}
\end{picture}
```

\picture
  ⟨*contenido del entorno*⟩
\endpicture

Macros (latex.ltx→P.1036, Ln: 5433) [8, p. 334], versión primitva del entorno `picture`→P.211.



Print Preview (Beta Test) ~ Presentación Preliminar

4.2. Atributos de dibujo

4.2.1. Unidad de longitud

\unitlength (default 1 pt)

Este comando [8, p. 334] es un registrador tipo *dimen*, que almacena la unidad de longitud (medida física) que representan los valores escalares de `picture`.

```
\unitlength=1cm
\begin{picture}(3,2)
    \put (0,0){\line(0,1){2}}
    \put (0,0){\line(1,0){3}}
    \put (0,0){\circle{.2}}
\end{picture}
```

Visibilidad:

28.45274pt
1.0pt

```
{
    \unitlength=1cm
    \the\unitlength\\
}
\the\unitlength
```

4.2.2. Grosor de línea

Estas declaraciones se pueden usar múltiples veces en una gráfica para cambiar el grosor de líneas particulares. `\thinlines` es el default.

\linethickness{⟨*grosor*⟩}

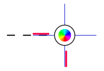
Establece la anchura de líneas verticales y horizontales. Este no tiene efecto para líneas inclinadas o círculos.

```
\begin{picture}(40,30)
    \linethickness{.4pt}
    \put (0,0){\line(0,1){30}}
    \linethickness{1.6pt}
    \put (0,0){\line(1,0){40}}
\end{picture}
```

\thinlines

Líneas delgadas.

```
\begin{picture}(90,30)
    \thinlines
```



Capítulo 5

Programación

Programación de alto y bajo nivel
—Programming in LaTeX and TeX

El capítulo trata la programación, es decir, la *creación* y gestión de nuevas funciones en el lenguaje (i.e., formato, según `\fmtname`→P.12) Plain TeX (versión 3.141592653) y LaTeX 2$_\varepsilon$ (versión 2017-04-15 o superior).

5.1. TeX

TeX es un lenguaje de macro, crear nuevas funciones significa crear macros (o, en terminología LaTeX, comandos).

5.1.1. Definiciones (Macros)

Definiendo nuevas macros [1, p. 198] [16, p. 109].

- `\def`→P.219, definición local
- `\edef`→P.223, definición local expandido
- `\gdef`→P.223, definición global
- `\xdef`→P.223, definición global expandido

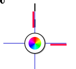
`\def`⟨*control sequence*⟩⟨*argument pattern*⟩`{`⟨*replacement text*⟩`}`

Comando primitivo TeX, define localmente la macro ⟨*control sequence*⟩ y su firma de ⟨*argument pattern*⟩ y a su invocación es expandido por ⟨*replacement text*⟩.

⟨*control sequence*⟩ es el identificador de la macro, formado según la sintaxis estándar (\⟨*csname*⟩→P.23); o puede ser un *token* de categoria 13.

⟨*argument pattern*⟩ es una expresión de tokens, *patrón de coincidencia*, que sirve para capturar los argumentos que recibe la macro. Los parámetros son de la forma `#1`, `#2` y asi susesivamente hasta un máximo de nueve. Si en la invocación los argumentos no cumplen el *patrón de coincidencia* se produce un error en tiempo de interpretación.

⟨*replacement text*⟩ es una serie de tokens, la especificación de expansión, cuando la macro es invocada se reemplaza los ⟨*argument pattern*⟩ `#1`, `#2`, etc. por sus valores pasados en la invocación.



Executing it: Lorem ipsumxyz.

```
\def\macro{Lorem ipsum}
Executing it: \macro xyz.
```

note que, en la expansión de una *secuencia de control* (véase \⟨*csname*⟩→P.23) los caracteres de espacio se depuran, la secuencia de invocación es:

`\macro` SP `xyz.`

Cuando la definición forma parte de un conjunto (i.e., ⟨*pre*⟩`\def...}`⟨*post*⟩) debe cuidar la interpretación de espacios (véase `\space`→P.29) en el *pre* y *post* definición.

y
x y.

```
\def\macro{Lorem ipsum} y

x
\def\macro{Lorem ipsum} y.
```

la secuencia de definición interpretado de esta última es:

`x` SP `\def\macro{Lorem ipsum}` SP `y.`

Buenas practicas de programación: con el fin de hacer mas legible el código puede usar saltos de línea (CR, U+000D y LF, U+000A) o identación (cuatro caracteres SP, U+0020), pero,

Executing it: ‘ Lorem ipsum ’.

```
\def\macro{
    Lorem ipsum
}
Executing it: `\macro'.
```

la secuencia de definición interpretado es:

`\def\macro{` SP `Lorem` SP `ipsum` SP `}`

El identificador ⟨*control sequence*⟩ puede ser dado como un simple token con categoria 13 (véase `\active`→P.32)

Executing it: ‘LoXXm iXsXm’.

```
\catcode`*=13
\def*{X}
Executing it: `Lo**m i*s*m'.
```

Macro ⟨*argument pattern*⟩ token:

#x (prefix operator)

El token # seguido de x, un número entre 1–9 se interpreta como parámetro de macro.

here is ‘A’BC
here is ‘AB’C

```
\def\macro#1{here is `#1'}
\macro ABC\\
\macro {AB}C
```

Interpretación de espacios despues de la captura de un argumento:

Lorem ipsum XYZ

```
\def\macro#1{Lorem ipsum}
\macro a XYZ
```

Interpretación de espacios entre ⟨*control sequence*⟩ y ⟨*argument pattern*⟩: los espacios despues de ⟨*control sequence*⟩ se depuran, hasta encontrar un token no espacio, desde ahi los espacios son parte del patrón de coincidencia para definir los ⟨*parámetros*⟩

here is “Lorem
”ipsum

```
\def\macro    #1   {here is ``#1''}
\macro Lorem\\
ipsum
```

la secuencia interpretada en la firma y en la invocación es:

```
\def\macro#1 SP {...}
\macro SP Lorem\\ SP ipsum.
```

Interpretación de otros caracteres en ⟨*argument pattern*⟩

here is “abc” and “def ghi” jkl mno.

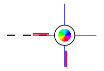
```
\def\macro#1 #2,{here is ``#1'' and ``#2''}
\macro abc def ghi, jkl mno.

%\macro abc def ghi jkl mno.       %%No compila, error de firma
```

note que la coma se captura como token (es parte de la firma de `\macro`) y por tanto no existe en la impresión. La última línea no cumple la firma de la macro, y produce un error, en el archivo log y la terminal:

```
Runaway argument?
def ghi jkl mno.
! Paragraph ended before \macro was complete.
```

first ‘A’, second ‘BC’, third ‘DEF’.G



Capítulo 6

Lenguaje de programación LATEX3

Programación de nivel extendido
—A LATEX3 programming environment

Habiendo establecido la programación estándar (§5); este capítulo cubre la programación de tercera generación extendidio del lenguaje LATEX.

6.1. Fundamentos

6.1.1. Especificación

File `expl3.ltx`

Este archivo 2021-02-18 (expl3.ltx→P.1038, Ln: 22) es el kernell de LATEX3, y declara la versión como un alias de `\fmtversion`→P.12.

```
\everyjob\expandafter{\the\everyjob
  \message{L3 programming layer <\ExplFileDate>}%
}
```

kernel: 2021-02-18
status=1

```
kernel: \ExplFileDate

\ExplSyntaxOn
status=\number\l__kernel_expl_bool
\ExplSyntaxOff
```

`\l__kernel_expl_bool` (variable booleano, modo lenguaje on=1, off=0). Metadatos del kernel:

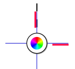
`\ExplFileDescription`,

`\ExplFileDate`,

`\ExplFileName`,

`\ExplFileExtension`,

`\ExplFileVersion`.



PRINT PREVIEW (BETA TEST) ~ PRESENTACIÓN PRELIMINAR

La especificación del lenguaje LATEX3 se desarrolla con base la implementación (expl3.ltx→P.1038) y la documentación oficial (source3.tex→P.1038) [31].
Fundamentos de la especificación, por ejemplo para el comando `\ExplSyntaxOn`→P.328:

1. Readme

   se da una noción en [4, p. 13], [32, p. 624] y [33, p. 14]; y en la documentación oficial se presenta en (source3body.tex→P.1038, ln: 271) [31, p. 3].

2. Source code

   Su código fuente reside en (expl3-code.tex→P.1038, ln: 275);

3. Implementation

   Describe el código fuente en (l3bootstrap.dtx→P.1038, ln: 620) [31, p. 269];

4. Documentation

   Documenta su sintaxis en (l3bootstrap.dtx→P.1038, ln: 158) [31, p. 6].

6.1.2. Paquete expl3

`\usepackage{expl3}`

Paquete versión 2021-02-18 (expl3.sty→P.1038, .pdf) [33], descrito como *Experimental LATEX3 programming layer (loader)*, define algunas macros y carga (expl3.sty→P.1038, Ln: 58) la fuente del código (expl3-code.tex→P.1038).

1. `\ProvidesExplPackage`→P.328
2. `\ProvidesExplClass`→P.328
3. `\ProvidesExplFile`→P.328

`\ProvidesExplPackage{`⟨*nombre*⟩`}{`⟨*fecha*⟩`}{`⟨*versión*⟩`}{`⟨*descripción*⟩`}`

Este comando (expl3.sty→P.1038, ln: 29) [31, p. 6] es similar al correspondiente `\ProvidesPackage`→P.304. Version identifiers, Issue 11 [4, p. 23]. Por ejemplo,

```
\ProvidesExplPackage{pkgproof}%
                    {2017/11/05}{2.6f}{Lorem ipsum for LaTeX3}
```

en el archivo log escribe:

```
Package: pkgproof 2017/11/05 v2.6f Lorem ipsum for LaTeX3
```

`\ProvidesExplClass{`⟨*nombre*⟩`}{`⟨*fecha*⟩`}{`⟨*versión*⟩`}{`⟨*descripción*⟩`}`

Este comando (expl3.sty→P.1038, ln: 31) [31, p. 6] es similar al correspondiente `\ProvidesClass`→P.306.

`\ProvidesExplFile{`⟨*nombre*⟩`}{`⟨*fecha*⟩`}{`⟨*versión*⟩`}{`⟨*descripción*⟩`}`

Este comando (expl3.sty→P.1038, ln: 33) [31, p. 6] es similar al correspondiente `\ProvidesFile`→P.303.

6.1.3. Entorno de programación LATEX3

`\ExplSyntaxOn`⟨*código*⟩`\ExplSyntaxOff`

Estos comandos (expl3-code.tex→P.1038, ln: 275) [31, p. 269, 6, 3] modifican la categoria de los caracteres `_` (U+005F) y `:` (U+003A) como "letras" (`\catcode`→P.22, 11) para que pueda escribir el ⟨*código*⟩ en lenguaje LATEX3.

⟨*código*⟩ interpretación de espacios y el token `~` (para ingresar un espacio).

Print Preview (Beta Test) ~ Presentación Preliminar

Capítulo 7

Composición con extensiones

Composición tipográfica extendida
—Typesetting extends, i.e., packages

La composición extendida cubre las capacidades de LaTeX estandar con extensiones. Al igual que la instrucción `import` o `using` para los lenguajes Java y C# respectivamente, el uso de una dependencia en LaTeX se declara mediante `\usepackage`→P.18.

7.1. Símbolos

7.1.1. Paquete `fourier-orns`

`\usepackage{fourier-orns}`

Este paquete v1.1 (2004/01/30, fourier-orns.sty→P.1055, .pdf) provee logos y ornaments (Tabla 7.1) en la codificación de fuente `U`, familia `futs`.

decoone ✖	`decoone {\fontencoding{U}\fontfamily{futs}\selectfont\char88}`

Tabla 7.1. Fourier-orns Package Commands Code Symbol

Glifo	Comando	Slot
€	`\eurologo`	69
⛔	`\noway`	65
⚠	`\danger`	66
💣	`\bomb`	76
☹	`\grimace`	77
⚲	`\textthing`	78
⚔	`\textxswup`	84
⚔	`\textxswdown`	85
✖	`\decoone`	88
൭	`\decothreeleft`	89

Tabla 7.1. (Continua de la página anterior)

Glifo	Comando	Slot
	\decothreeright	90
	\decofourleft	91
	\decofourright	92
	\decosix	93
	\decotwo	97
	\floweroneleft	98
	\floweroneright	99
	\starredbullet	100
	\leafNE	102
	\leafleft	103
	\leafright	104
	\aldinesmall	106
	\aldineleft	109
	\aldineright	110
	\aldine	111
	\lefthand	116
	\righthand	117

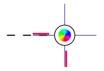
7.1.2. Paquete epiolmec

`\usepackage{epiolmec}`

Este paquete v1.0 (2003/11/05, epiolmec.sty→P.1054, .pdf) provee glifos conocidos de Epi-Olmec, en la codificación de fuente LEO.

La Tabla 7.2, conforme con GlyphAccessCommands.pdf→P.1054, lista los comandos creados con \DeclareTextSymbol .

Por ejemplo, \EOSprinkle, selección {\fontencoding{LEO}\fontfamily{cmr}\selectfont\char139} .



Capítulo 8

Composición gráfica con Ti*k*Z & PGF

Composición gráfica vectorial extendida
—Extended picture environment

8.1. Fundamentos

8.1.1. Plataforma

Ti*k*Z o (por sus siglas en alemán) “Ti*k*Z ist *kein* Zeichenprogramm” [58, §11, p. 124] es una plataforma PGF (una serie de paquetes) para la composición de gráficos, un “lenguaje de descripción de gráficos” (*no* un editor gráfico) [59, p. 496] desarrollado por Till Tantau desde el 2003.

8.1.2. Versión

Ti*k*Z (Ti*k*Z ist *kein* Zeichenprogramm) 2005 to 2007.

Mapeo del lanzamiento de las distintas versiones estables (CHANGELOG.md→P.1060) agrupadas por generación, conforme especificación `\pgfversion`→P.789.

Versiones: 1.00 (2005-10-23); 1.01 (2005-11-16); 1.09 (2006-10-11); 1.10 (2006-10-26); 1.18 (2007-01-18).

Ti*k*Z 2 (Segunda generación) 2008 to 2010.

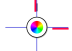
Versiones: 2.00 (2008-02-20); 2.10 (2010-10-25).

Ti*k*Z 3 (Tercera generación) 2013 to 2015.

Versiones: 3.0.0 (2013-12-20); 3.0.1 (2015-08-07); 3.0.1a (2015-08-07) [60].

Ti*k*Z 3.1 (Tercera generación extendida) 2019 to 2024.

Versiones: 3.1 (2019-01-05); 3.1.1 (2019-02-02); 3.1.2 (2019-04-04); 3.1.3 (2019-05-09); 3.1.4 (2019-07-12); 3.1.4a (2019-07-16); 3.1.4b (2019-08-03); 3.1.5 (2019-12-19); 3.1.5a (2019-12-21); 3.1.5b (2020-01-08); 3.1.6 (2020-09-28); 3.1.6a (2020-10-01); 3.1.7 (2020-11-21); 3.1.7a (2020-12-01); 3.1.8 (2020-12-25); 3.1.8b (2020-12-27); 3.1.8a (2020-12-27); 3.1.9 (2021-03-02); 3.1.9a (2021-05-15) [58] [61]; 3.1.9a-10-gafa8dcc6 (2021-07-02) [62]; 3.1.10 (2023-01-13) [63], [64].



8.1.3. Paquete tikz

`\usepackage{tikz}`

Este paquete v3.0.1a (2015/08/07, tikz.sty→P.1060, pgfmanual.pdf) [58, p. 128] provee un lenguaje declarativo y expresivo de alto nivel para crear gráficos vectoriales de forma nativa para LaTeX.

Dependencias del paquete: pgf→P.641 y pgffor→P.783.

Note que esta implícito que este paquete es necesario para todos los dibujos de este capítulo. También se da por implícito que este paquete indirectamente requiere los siguientes paquetes: xcolor→P.417; everyshi→P.559; graphicx→P.452; keyval→P.504; graphics→P.451; trig→P.516.

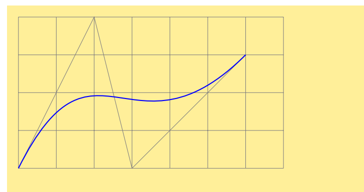
8.1.4. Modo dibujo

`\begin{tikzpicture}[`⟨*opciones*⟩`]`
⟨*contenido del entorno*⟩
`\end{tikzpicture}`

Este entorno [58, p. 128] se usa para dibujar un gráfico vectorial. En el ⟨*contenido del entorno*⟩ se activa el modo dibujo, en el, se escriben las instruciones (una serie de secuencia de tokens) que sirven para producir la figura formateada con las ⟨*opciones*⟩ especificadas.

⟨*contenido del entorno*⟩: es una serie de comandos de dibujo (§8.1.5) separados por punto y coma. Cada comando recibe, como argumento, una serie de funciones de dibujo (§8.3) que sirven para construir una figura (forma geométrica).

⟨*opciones*⟩: es una serie de atributos de dibujo (§8.4) separados por comas. Cada atributo, es un argumento (opcional) de una función de dibujo, modifica el valor de un estilo (o formato) predeterminado para la forma geométrica (o toda la figura) según la visibilidad en el grupo.



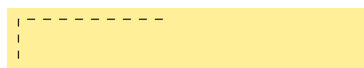
```
\begin{tikzpicture}[scale=.5]
  \draw[help lines]    (0,0)        grid          (7,4);
  \draw[help lines]    (0,0) -- (2,4)  -- (3,0) -- (6,3);
  \draw[blue] (0,0) .. controls (2,4) and (3,0) .. (6,3);
\end{tikzpicture}
```

`\tikzpicture[`⟨*opciones*⟩`]`
⟨*contenido del entorno*⟩
`\endtikzpicture`

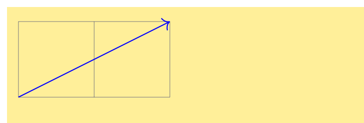
Versión primitva del entorno tikzpicture→P.570 (tikz.code.tex→P.1060, line 1629 y 1647) [58, p. 130].

`\tikz[`⟨*opciones*⟩`]`⟨*comandos de dibujo*⟩

Esta macro (tikz.code.tex→P.1060, line 1676) [58, p. 130], similar al entorno tikzpicture→P.570 pero, util cuando solo requieren pocas líneas de ⟨*comandos de dibujo*⟩.



```
\tikz[dashed] \draw (0,0) |- (15:2);
```



```
\tikz { \draw[help lines]   (0,0) grid (2,1);
        \path[draw=blue,->] (0,0)  --  (2,1);
}
```

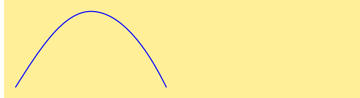
8.1.5. Comandos de dibujo

`\path[`⟨*opciones*⟩`]`⟨*funciones de dibujo*⟩`;`

Esta macro [58, p. 153] dibuja una serie de ⟨*funciones de dibujo*⟩.

⟨*funciones de dibujo*⟩: define el gráfico mediante una serie de *operaciones* (o, en terminología SVG, comandos) de ruta. Se prefiere el término *función* conforme su relación directa con las funciones matemáticas de *geometría diferencial*.

Notese el '`;`' al final, i.e., una ⟨*función de dibujo*⟩ termina en punto y coma.



```
\tikz
  \path[draw=blue] (0,0) sin (1,1) parabola (2,0);
```

`\draw`

Abreviación de `\path[draw]`, véase atributo `draw`→P.610.

`\fill`

Abreviación de `\path[fill]`, véase atributo `fill`→P.614.

`\filldraw`

Abreviación de `\path[fill,draw]`.

`\pattern`

Abreviación de `\path[pattern]`, véase atributo `pattern`→P.617.

`\shade`

Abreviación de `\path[shade]`, véase atributo `shade`→P.618.

`\shadedraw`

Abreviación de `\path[shade,draw]`.

`\clip`

Abreviación de `\path[clip]`.

`\graph`

Abreviación de `\path graph`, véase función `graph`→P.605.

`\useasboundingbox`

Abreviación de `\path[use as bounding box]`, véase atributo `use as bounding box`→P.633.

`\node`

Abreviación de `\path node` [58, p. 227], véase función `node`→P.596.

`\matrix`

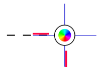
Abreviación de `\path node[matrix]`, véase atributo `matrix`→P.628.

`\coordinate`

Abreviación de `\path coordinate` [58, p. 227], véase función `coordinate`→P.600.

`\pic`

Abreviación de `\path pic` [58, p. 262], véase función `pic`→P.602.



Capítulo 9

Proyecto MDTZIØFEX

Contribución, experiencia e investigación
—Advancing research. Administrative experience

Este proyecto es el cimiento de la presente obra. Provee interfaces para su escritura (a nivel de código), regula su estructura (a nivel renderizado), monitorea su evolución (estadísticas de progreso), y realiza investigación avanzada (lógica, matemática y programación) para enriquecer el contenido del documento principal. Técnicamente la obra le debe su existencia a este proyecto.

9.1. Introducción

El proyecto MDTZIØFEX se crea para dar soporte a las nuevas obras literarias de MDT496 ESTUDIOS.

9.1.1. Logos

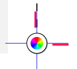
El logo se define por: **MDT** (MDT496 ESTUDIOS); **ZIO** (mathematical); **FEX** (programming).

```
% MDTZiOFex
%    760                      496 octal
% 111110000                   496 binario
%     01F0                    496 hexadecimal
%        ex(te|pa)nds         Conciso y expresivo
```

`\mdtziofex`

Este comando robusto (mdtziofex-main.tex→P.900, Ln: 12) expande el logo del proyecto MDTZIØFEX.

```
\protected\gdef\mdtziofex{\leavevmode\hbox{\m@th
    \if b\expandafter\@car\f@series\@nil\boldmath\fi
    MDT\kern-.26em%
    {\usefont{OML}{cmm}{b}{it}\char`7}%
    \lower.19444em\hbox to 0pt{%
          \hss{\vrule width.7em height.05em}\hss}%
    $\mathchar379$%
    \lower.19444em\hbox{\usefont{OMS}{cmsy}{b}{n}\char59}%
    F\kern-.1667em{\usefont{OT1}{cmtt}{m}{sc}ex}}}
```



PRINT PREVIEW (BETA TEST) ~ PRESENTACIÓN PRELIMINAR

MDTziøFex	`\mdtziofex`

inspirado en la implementación de logos como: `\TeX`→P.11, `\LaTeX`→P.11, `\LaTeXe`→P.12, `\AmS`→P.404 y `\AMS`→P.547, es escalable en modo párrafo y matemático.

MDTziøFex	`{\huge\bfseries\mdtziofex}`

9.1.2. Terminología Unicode

Para evitar ambigüedad de los caracteres que se escriben en el código fuente, se emplea como referencia el estándar Unicode [14] para identificar un carácter visible (e.g., β (U+03B2)) y un carácter de control (e.g., CR (U+000D)).
Por ejemplo, cuando copia el siguiente código fuente

Lorem' ipsum Lorem' ipsum	`Lorem' ipsum\par` `Lorem’ ipsum`

debe notar[1] que en el código fuente se teclea los caracteres `'` (U+0027) y `’` (U+2019). Las propiedades Unicode [14, p. 869] de estos se muestran en la Tabla 9.1.

Tabla 9.1. Análisis de codificación de caracteres Unicode

Glifo	Unicode	Name	Type	Block
'	0027	APOSTROPHE	Other Punctuation (Po)	Basic Latin (0000)
’	2019	RIGHT SINGLE QUOTATION MARK	Final Quote Punctuation (Pf)	General Punctuation (2000)

9.2. Libreria para documentar macro

MDTziøFex —*Macro Documentation Support*

Proyecto MDTziøFex, módulo *Macro Documentation Support* (`texspecs`→P.822). Este módulo provee interfaces (macros, entornos, etc.) para escribir la documentación de una macro. Estas son implementadas cuidadosamente, con soporte hasta el nivel de lenguaje LaTeX3.

9.2.1. Abreviaturas

`\cmd`⟨*control sequence*⟩

Command ⟨*control sequence*⟩. `\cmd` es un comando robusto.

prints `\macro`.	`prints \cmd\macro.`

Soporta comandos LaTeX3

prints `\cs_new:Npn`.	`prints \cmd{\cs_new:Npn}.`

[1] El compilado produce el mismo glifo, esto es tema de codificación del archivo (UTF-8, véase `inputenc`→P.549) y fuente (OT1, véase `fontenc`→P.553) que no se trata aquí.

9.7. Investigación

9.7.1. The LaTeX Language Specification (Synthesis)

Este manual tiene como objetivo general de especificar los comandos del lenguaje LaTeX; por herencia se especifica los comandos primitivos de TeX; finalmente, por correlación (i.e., comandos en comun) se especifica algunas funciones del formato "plain TeX".

Tabla 9.2. The LaTeX Language Specification (Synthesis)

Lenguaje	Progreso	comandos especificados
TeX primitive→P.906	322/322	100.0 %
Plain TeX→P.910	518/518	100.0 %
LaTeX 2$_\varepsilon$→P.918	974/974	100.0 %
LaTeX3→P.932	53/1157	4.0 %
Ti*k*Z→P.950	283/777	36.0 %
—***Consolidado***—	1684/3282	51.0 %

La Tabla 9.2, conforme la §9.7.2→P.961 (*The LaTeX Specification Request*), detalla el progreso de avance de la cantidad de comandos especificados para un nivel de lenguaje. Las secuencias de control que implementan según el nivel del lenguaje se describen en la siguiente lista:

1. Síntesis lenguaje TeX primitive, Tabla 9.3 (322 comandos).

   Comandos primitivos del lenguaje TeX, donde Knuth [1, p. 10] indica son 300 comandos, pero alguno de ellos fueron redefinidos por los diferentes lenguajes que heredan de el.

2. Síntesis lenguaje Plain TeX, Tabla 9.4 (518 comandos).

   'Macros Básicos' originales [8, p. 14] implementadas por Knuth [1, p. 10], que indica que son cerca de 600 comandos, y descritas en su Apéndice B [1, p. 339].

3. Síntesis lenguaje LaTeX 2$_\varepsilon$, Tabla 9.5 (974 comandos).

   Las macros, comandos (robustos) y entornos primitivos del lenguaje LaTeX 2$_\varepsilon$ [8].

   La tabla contiene todos los comandos sin discriminación (LaTeX 2.09 y LaTeX 2$_\varepsilon$), sin embargo LaTeX 2$_\varepsilon$ es una actualización [10] [2, p. 227], por tanto provee comandos nuevos y deja obsoleto a otros. Por ejemplo, provee el comando `\tt` de plain TeX y remueve a `\mho` de LaTeX 2.09.

4. Síntesis lenguaje LaTeX3, Tabla 9.6 (1157 comandos).

   La tabla, aun en desarrollo, contiene solo algunos de los identificadores (funciones, variables y macros) del lenguaje LaTeX3.

5. Síntesis lenguaje Ti*k*Z, Tabla 9.7 (777 comandos).

   La tabla contiene todos los comandos considerando: macros (textoken como `\pgflinewidth`); comandos matemáticos (como `\pgfmathpi`); entornos primitivos (como `\tikzpicture`); y redefiniciones (como `\fill`).

   El rastreo de una especificación oficial inicia por el índice [58, p. 1272]. En consideración se tienen macros sin especificación oficial como `\pgfmathresult`; con especificación oficial como `\pgfmathparse` [58, p. 1028].

Tabla 9.3. Síntesis lenguaje TEX primitive (322 comandos)

TEX primitive	TEX primitive
\above→P.206	\mathchardef→P.233
\abovedisplayshortskip→P.177	\mathchoice→P.182
\abovedisplayskip→P.177	\mathclose→P.302
\abovewithdelims→P.207	\mathcode→P.184
\accent→P.248	\mathinner→P.195
\adjdemerits→P.254	\mathop→P.302
\advance→P.233	\mathopen→P.302
\afterassignment→P.246	\mathord→P.302
\aftergroup→P.247	\mathpunct→P.302
\atop→P.206	\mathrel→P.302
\atopwithdelims→P.207	\mathsurround→P.176
\badness→P.253	\maxdeadcycles→P.253
\baselineskip→P.34	\maxdepth→P.255
\batchmode→P.316	\meaning→P.250
\begingroup→P.25	\medmuskip→P.178
\belowdisplayshortskip→P.177	\message→P.313
\belowdisplayskip→P.177	\mkern→P.177
\binoppenalty→P.46	\month→P.154
\botmark→P.162	\moveleft→P.227
\box→P.243	\moveright→P.227
\boxmaxdepth→P.255	\mskip→P.178
\brokenpenalty→P.46	\multiply→P.234
\catcode→P.22	\muskip→P.232
\char→P.21	\muskipdef→P.232
\chardef→P.233	\newlinechar→P.255
\cleaders→P.42	\noalign→P.88
\closein→P.154	\noboundary→P.252
\closeout→P.154	\noexpand→P.251
\clubpenalty→P.46	\noindent→P.36
\copy→P.243	\nolimits→P.196
\count→P.232	\nonscript→P.252
\countdef→P.232	\nonstopmode→P.316
\cr→P.86	\nulldelimiterspace→P.255
\crcr→P.87	\nullfont→P.252